一种求解函数优化问题的改进鲸鱼优化算法 *

刘 亮[a, b]，何 庆[a, b†]
(贵州大学 a. 大数据与信息工程学院; b. 贵州省公共大数据重点实验室，贵阳 550025)

摘 要：为提高鲸鱼优化算法求解复杂函数优化问题的性能，提出一种基于自适应参数及小生境技术的改进鲸鱼优化算法。首先，引入自适应概率阈值协调算法的全局探索及局部开发能力；其次，利用自适应位置权重对鲸鱼位置更新公式进行调整，提高算法的收敛速度及寻优精度；最后，采用预选择小生境技术，避免算法出现早熟收敛的现象。通过对 12 个典型基准测试函数的仿真表明，改进算法的寻优精度和收敛速度均较对比算法有明显提升，证明了提出的改进策略能有效提高鲸鱼优化算法求解复杂函数优化问题的性能。

关键词：函数优化；鲸鱼优化算法；自适应参数；小生境
中图分类号：TP301.6 doi: 10.19734/j.issn.1001-3695.2018.11.0726

Improved whale optimization algorithm for solving function optimization problems

Liu Liang[a, b], He Qing[a, b†]
(*a. College of Big Data & Information Engineering, b. Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guiyang* 550025, *China*)

Abstract: In order to improve the performance of whale optimization algorithm for solving complex function optimization problems, this paper proposed an improved whale optimization algorithm based on adaptive parameters and niche technology. Firstly, the algorithm introduced an adaptive probability threshold to coordinate the exploration and exploitation ability. Then, the algorithm used adaptive position weights to adjust the whale position update formula to improve the convergence speed and search precision. Finally, the algorithm used preselection niche technology to avoid premature convergence. The results on 12 typical benchmark functions shows that the improved algorithm has faster convergence speed and higher search precision than other comparison algorithms. It proves that the improvement strategy can effectively improve the performance of the whale optimization algorithm for solving complex function optimization problems.

Key words: function optimization; whale optimization algorithm; adaptive parameters; niche

0 引言

鲸鱼优化算法（whale optimization algorithm, WOA）[1]是由Mirjalili等人于2016年提出的一种模拟座头鲸捕猎行为的新型群体智能优化算法，该算法的主要思想是通过模仿鲸鱼的捕食行为实现对目标问题的求解[2]。经相关实验证明，WOA 无论是算法收敛速度还是寻优精度都要优于粒子群优化算法（PSO）[3]和引力搜索算法（GSA）[4]等经典算法，且目前已将其成功应用于水资源优化配置[5]、最优控制[6]以及特征选择[7]等领域，但与其他群体优化算法相类似，传统 WOA 仍然存在早熟收敛、收敛速度慢以及无法找到全局最优解的问题。因此，为了提高 WOA 算法的收敛速度和寻优精度，近年来国内外学者分别从不同的角度对 WOA 算法进行了改进，如 Abdel-Basset 等人[8]利用 Lévy 飞行以及逻辑混沌映射来替换和确定 WOA 中的系数向量 $\vec{C}$ 以及切换概率 p，提出一种改进的 WOA 算法，并通过实验证明了算法的有效性及优越性；Sayed 等人[9]将混沌搜索引入到 WOA 的迭代搜索中，提出一种 CWOA 算法，利用混沌映射来定义 WOA 中的 $\vec{A}$、$\vec{C}$、l 及 p 等参数，通过与 WOA 以及其他 10 种优化算法进行比较，证明了该方法可显著提高 WOA 的性能；郭振洲等人[10]通过对鲸鱼位置进行柯西变异，并引入自适应权重系数，提出了一种 WOAMC 算法，通过实验证明，该算法在收敛精度和算法稳定性上都要优于传统 WOA 算法；龙文等人[11]提出了一种 IWOA 算法，通过非线性变化的收敛因子更新公式协调算法的探索和开发能力，并对当前最优个体执行多样性变异操作，实验证明改进算法的寻优精度和收敛速度均明显提高；王坚浩等人[12]采用混沌反向学习以及非线性混沌扰动的改进方法，提出一种基于混沌搜索策略的改进算法，并对多个基准函数进行测试，证明了该算法相较于传统 WOA 算法各项性能均有所提高。可见，目前针对 WOA 存在的问题已有了许多先进的研究成果，但如何进一步提升算法的寻优精度以及收敛速度仍需进行更深入的研究。

本文首先对 WOA 中鲸鱼捕食策略选择的概率阈值进行修改，采用自适应参数代替原有的固定阈值，平衡算法在迭代过程中的全局开发和局部探索能力；其次将自适应参数作为权重系数，调整鲸鱼位置更新公式，提高算法的寻优精度和收敛速度；最后结合基于预选择机制的小生境算法，在保证鲸鱼朝着“猎物”移动的同时，维持种群的多样性，避免算法陷入局部最优。通过对 12 个基准测试函数的仿真结果表明，采用上述三种改进策略能够显著地提高算法的寻优精度

收稿日期：2018-11-02；**修回日期：**2018-12-03 **基金项目：**贵州省科技计划项目重大专项资助项目（黔科合重大专项字［2018］3002）；贵州省公共大数据重点实验室开放课题（2017BDKFJJ004）；贵州省教育厅青年科技人才成长项目（黔科合 KY 字［2016］124）；贵州大学培育项目（黔科合平台人才［2017］5788）

作者简介：刘亮（1994-），男，贵州六盘水人，硕士研究生，主要研究方向为数据挖掘、进化计算；何庆（1982-），男（通信作者），贵州贵阳人，副教授，博士，主要研究方向为大数据应用、人工智能（qhe@gzu.edu.cn）.

和收敛速度。

1　鲸鱼优化算法

相较于其他种类鲸鱼，座头鲸具有独特的捕食方式，即泡泡网补食法，如图 1 所示。WOA 算法即是源于座头鲸泡泡网捕食行为的启发而衍生出的一种新型仿生群体智能算法。因此，根据座头鲸的泡泡网捕食法的特点，WOA 算法主要可分为包围猎物、泡泡网攻击以及搜索猎物三个不同阶段。

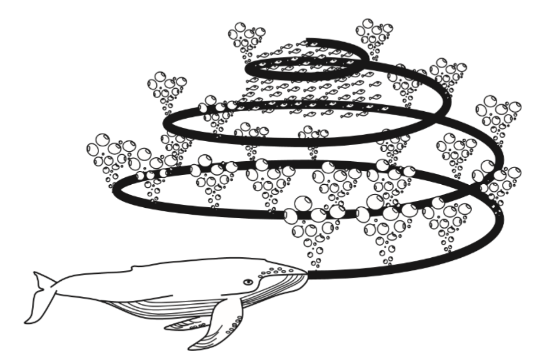


图 1　泡泡网捕食法

Fig. 1　Bubble-net attacking method

1.1　包围猎物

鲸鱼在捕食的过程中，首先需确定猎物的位置才能包围捕获猎物，在面对实际优化问题时，由于搜索空间中猎物的位置往往是未知的，所以 WOA 算法假设当前种群中的最优解为目标猎物；在确定猎物之后，种群中的其他鲸鱼将根据当前猎物的位置来更新自身位置，如下所示[13]：

$$\vec{D}=\left|\vec{C}\cdot\vec{X}^*(t)-\vec{X}(t)\right| \tag{1}$$

$$\vec{X}(t+1)=\vec{X}^*(t)-\vec{A}\cdot\vec{D} \tag{2}$$

其中：t 表示当前迭代次数；$\vec{X}^*$ 为当前群体中最优解的位置向量；$\vec{X}(t)$ 表示鲸鱼当前所在位置；$\vec{A}\cdot\vec{D}$ 表示包围步长；$\vec{A}$ 和 $\vec{C}$ 为系数向量并按下式定义：

$$\vec{A}=2\vec{a}\cdot r_1-\vec{a} \tag{3}$$

$$\vec{C}=2\cdot r_2 \tag{4}$$

其中：r_1 与 r_2 为[0,1]的随机数；$\vec{a}$ 为随迭代次数增加取值由 2 线性递减为 0 的控制参数，可表示为

$$a=2-2t/Max_iter \tag{5}$$

其中：Max_iter 为最大迭代次数。

1.2　泡泡网攻击

座头鲸的泡泡网觅食特点是在收缩的包围圈内沿着螺旋路径朝着猎物移动，因此 WOA 算法设计了收缩包围机制以及螺旋更新位置两种策略来模拟座头鲸独特的泡泡网捕食行为。

a) 收缩包围机制通过降低式(3)中的 a 值实现。由式(3)可知，$\vec{A}$ 的值随 a 的降低而降低，当 a 由 2 线性递减为 0 时，$\vec{A}$ 的取值为[-a, a]；因此，当 $\vec{A}$ 在[-1,1]内取值时，更新位置后的鲸鱼必定处于原始位置与猎物之间，即使得每条鲸鱼由原来的位置向猎物靠近，完成对猎物的包围。

b) 螺旋更新位置首先需计算鲸鱼与猎物之间的距离，然后在鲸鱼与猎物之间创建螺旋方程以模仿座头鲸的螺旋运动状态，如式（6）所示[14]。

$$\vec{X}(t+1)=\vec{D}'\cdot e^{bl}\cdot\cos(2\pi l)+\vec{X}^*(t) \tag{6}$$

其中：$\vec{D}'=\left|\vec{X}^*(t)-\vec{X}(t)\right|$ 为鲸鱼与猎物之间的距离；b 为用于定义螺旋形状的常数，本文取 b=1；l 为[-1,1]间的随机数。

由于鲸鱼在收缩包围圈的同时，还需沿着螺旋路径向猎物移动，为了模拟这种同步过程，WOA 算法假设鲸鱼在进行狩猎的过程中选择两种策略的概率都为 0.5，其数学模型可表示为

$$\vec{X}(t+1)=\begin{cases}\vec{X}^*(t)-\vec{A}\cdot\vec{D} & if\ p<0.5\\ \vec{D}'\cdot e^{bl}\cdot\cos(2\pi l)+\vec{X}^*(t) & if\ p\ge 0.5\end{cases} \tag{7}$$

1.3　搜索猎物

与收缩包围机制相反，在搜索猎物阶段，当 $\vec{A}$ 满足 $|\vec{A}|>1$ 时，鲸鱼通过彼此的位置随机地搜索猎物，因此不再选择猎物来更新自身位置，而是在群体中随机地选择一个个体来代替原猎物的作用，迫使鲸鱼远离猎物所在位置，以增强 WOA 算法的全局寻优能力。其数学模型表示如下：

$$\vec{D}=\left|\vec{C}\cdot\vec{X}_{rand}-\vec{X}\right| \tag{8}$$

$$\vec{X}(t+1)=\vec{X}_{rand}-\vec{A}\cdot\vec{D} \tag{9}$$

其中：$\vec{X}_{rand}$ 即为从当前群体中随机选择鲸鱼的位置向量。

2　基于自适应参数及小生境的改进鲸鱼优化算法

为提高鲸鱼优化算法求解复杂函数优化问题的性能，本文结合三种改进思路，提出一种基于自适应参数及小生境技术的改进鲸鱼优化算法（whale optimization algorithm based on adaptive parameters and niche，APN-WOA）。

2.1　自适应概率阈值

WOA 算法在泡泡网攻击阶段，为了模拟收缩包围机制以及螺旋更新位置两种策略的同步进行，设置鲸鱼选择其中任意一种策略的概率都为 50%，即概率阈值为 0.5，如式（7）所示，通过在[0,1]之间随机生成的 p 值与概率阈值相比较，来选择觅食策略。在算法的迭代过程中，这种等概率的策略选择方式可能会使得鲸鱼无法选择适合当前群体的捕食策略，使得算法出现收敛速度慢、陷入局部最优等问题；因此本文引入自适应参数代替原有的概率阈值，该自适应阈值可随着算法的迭代在[0,1]之间变化，使得鲸鱼在不同时期有较大的概率选择到适合当前群体的捕食策略，从而协调算法的全局探索和局部开发能力，提高算法收敛速度。其数学表达式如下：

$$Adaptive_p=1-[\frac{1}{\lambda+\mu}\cdot(\lambda\cdot\frac{t^{\lambda}}{Max_iter^{\lambda}}+\mu\cdot\frac{t^{\mu}}{Max_iter^{\mu}})] \tag{10}$$

其中：t 为当前迭代次数；Max_iter 为最大迭代次数；λ、μ 为控制参数，$\lambda=3$，$\mu=2$。因此，可将式（7）改写为

$$\vec{X}(t+1)=\begin{cases}\vec{X}^*(t)-\vec{A}\cdot\vec{D} & if\ p<Adaptive_p\\ \vec{D}'\cdot e^{bl}\cdot\cos(2\pi l)+\vec{X}^*(t) & if\ p\ge Adaptive_p\end{cases} \tag{11}$$

由式（11）可知，在算法迭代初期，自适应阈值较大，使得鲸鱼有较大的概率选择收缩包围机制；在算法迭代后期，自适应阈值较小，将有较大概率选择螺旋位置更新，因此，鲸鱼由原来两种方式同步进行转变为先收缩包围再进行螺旋位置更新，使得其更快地靠近猎物，提升算法收敛速度。

2.2　自适应位置权重

由式（2）（7）（9）可知，在 WOA 算法的位置更新过程中，除相应的控制参数外，影响鲸鱼进行位置更新的主要因素为猎物位置向量 $\vec{A}$ 以及种群中随机选取的鲸鱼的位置向量 $\vec{X}_{rand}$，但 WOA 未考虑到在算法迭代过程中猎物引导鲸鱼进行位置更新的引导力可能存在差异，以及不同阶段内种群中随机选择的鲸鱼与猎物之间关系也是不同的；因此，受文献[15，16]的启发，本文将自适应参数作为权重系数，结合式（11）对 WOA 的位置更新进行调整，定义如下：

$$\omega=\frac{1}{\lambda+\mu}\cdot\left(\lambda\cdot\frac{t^{\lambda}}{Max_iter^{\lambda}}+\mu\cdot\frac{t^{\mu}}{Max_iter^{\mu}}\right) \tag{12}$$

$$\vec{X}(t+1)=\omega\cdot\vec{X}^{*}(t)-\vec{A}\cdot\vec{D},|\vec{A}|<1,p<Adaptive_p \tag{13}$$

$$\vec{X}(t+1)=\omega\cdot\vec{X}_{rand}(t)-\vec{A}\cdot\vec{D},|\vec{A}|\geq 1,p<Adaptive_p \tag{14}$$

$$\vec{X}(t+1)=\vec{D}'\cdot e^{bl}\cdot\cos(2\pi l)+(1-\omega)\cdot\vec{X}^{*}(t),p\geq Adaptive_p \tag{15}$$

其中：$\lambda=3$，$\mu=2$，ω 在[0,1]间取值。当 ω 随迭代次数增加而增加时，表明经过每次迭代所选择的猎物也即是当前种群的最优解随着自身适应度的提高，其对种群中的鲸鱼产生的吸引力也越强；同样，随着迭代的进行，种群中随机选择的鲸鱼所传达的信息可信度也会随之增高，因此，在式（13）（14）中权重系数随着迭代次数增加而增加，根据自适应的权重变化，使得鲸鱼能够更准确地找到猎物，从而提高算法收敛速度和寻优精度；但在算法迭代后期进行螺旋位置更新时，鲸鱼将向猎物靠近，此时应采用较小的权重系数，如式（15）所示，使得鲸鱼更新位置的同时能够更好地寻找猎物周围是否存在更优解，以此提高算法局部开发能力。

2.3　预选择小生境技术

生物学中，小生境是指特定环境下的一种生存环境。最早使用小生境概念求解优化计算问题是将其与遗传算法相结合，使得遗传算法中的个体在一个特定的生存环境中进化，以避免算法陷入局部最优；而目前小生境的实现方式包括有基于预选择机制、基于排挤度以及基于共享函数等方式；本文将基于预选择的小生境思想与 WOA 相结合，以保证 WOA 的种群多样性，增强算法的全局寻优能力。

基于预选择小生境的主要思想是仅当新产生的子代个体适应度超过其父代个体适应度时，所产生出的子代才能替换其父代遗传到下一代中，由于子代父代结构相似，所以被替换的只是一些相似的个体，从而维持了种群多样性，造就小生境的进化环境[17]。因此，本文将算法迭代前后每条鲸鱼的位置分别存储在记忆矩阵 $\boldsymbol{M}_L$ 和 $\boldsymbol{M}_N$ 的行向量中，若 $\boldsymbol{M}_L$ 第 i 行所代表的鲸鱼的适应度优于 $\boldsymbol{M}_N$ 中第 i 行所代表鲸鱼的适应度，则将 $\boldsymbol{M}_N$ 的第 i 行以 $\boldsymbol{M}_L$ 中的第 i 行进行替换，即若位置更新后的鲸鱼适应度较优则保留；否则使该鲸鱼返回原来的位置，避免鲸鱼全部聚集在某个局部最优点，从而维持种群多样性，提高算法的全局寻优能力。

综上所述，APN-WOA 算法流程如图 2 所示。

3　实验仿真与分析

算法仿真测试均在 Intel(R) Core(TM) i5-6500 CPU、3.2 GHz 主频 8 GB 内存以及 Windows 7（64 位）操作系统的计算机上实现，编程软件为 MATLAB R2015b。为验证本文所提出的 APN-WOA 算法的有效性，引入 12 个典型基准测试函数进行测试，如表 1 所示，其中 F_1~F_6 为连续单模函数，F_7~F_{12} 为复杂非线性多模函数。

本文对算法的测试主要包括以下三个部分：a)在相同的种群大小、迭代次数条件下，比较改进算法与传统 WOA 算法以及灰狼优化算法（grey wolf optimizer，GWO）[18]在基准测试函数上的算法的收敛速度和寻优精度，证明本文所提出的 APN-WOA 算法的有效性；b)对不同策略下的改进算法进行独立测试，根据测试结果分析不同改进策略对算法性能的影响；c)通过与参考文献中其他改进算法作对比，证明 APN-WOA 算法相对于其他最新的改进 WOA 算法仍具有较强的竞争性。

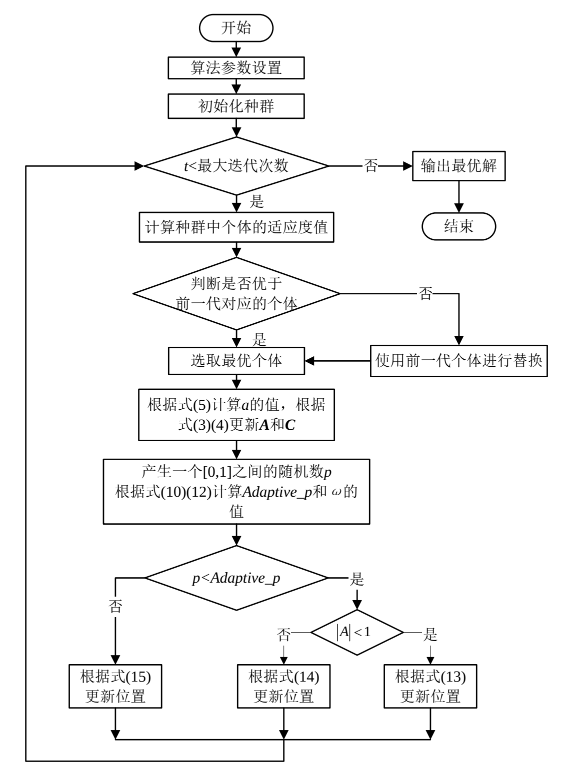


图 2　APN-WOA 算法流程图

Fig. 2　Algorithm flow chart of APN-WOA

3.1　算法性能测试

在不同维度（30/200/500）的搜索空间测试 APN-WOA 算法的性能，设置种群规模为 30，最大迭代次数为 500 次，分别将 GWO、WOA 以及 APN-WOA 三种算法独立运行 30 次，从最优解的均值以及标准差两个方面来衡量算法性能的差异，并引用文献[16]中的几组数据进行对比，测试结果如表 2 所示（“—”表示参考文献未给出）。为了更直观地反映 APN-WOA 算法的性能，图 3~5 给出了 30 维、200 维和 500 维搜索空间中算法对 12 个基准测试函数的优化收敛曲线。

由表 2 可知，APN-WOA 算法在不同维度下针对 12 个基准测试函数的寻优精度相较于传统 WOA 算法均有着明显提升；其中对函数 F_1、F_3、F_8 及 F_{10} 均取得了理论最优值，对函数 F_2、F_4 和 F_9 所求最优解的标准差也达到 0，接近其理论值；并且相较于 GWO 算法，APN-WOA 算法的寻优精度几乎全部优于 GWO 算法，仅在 30 维度下对函数 F_5 以及 200 维度下对函数 F_{12} 的优化上略微低于 GWO，但寻优精度仍在同一量级；而由图 3（e）（l）可知，APN-WOA 与 GWO 算法在相同量级的寻优精度情况下，APN-WOA 的收敛速度明显比 GWO 算法更快；并且根据不同维度下算法的优化收敛曲线对比可知，对于 12 个基准测试函数，APN-WOA 算法无论是在 30 维、200 维还是 500 维的搜索空间中，算法收敛速度相较于传统 WOA 算法以及 GWO 算法的均有明显提升，尤其对函数 F_1、F_2、F_3、F_4、F_6、F_8、F_9、F_{10} 提升效果最为显著；此外，相较于 MS-WOA 算法，APN-WOA 算法仅在 F_{11}，F_{12} 的优化上略劣于 MS-WOA 算法，但均在同一量级；而对于其他函数，APN-WOA 无论是在收敛速度还是寻优精度方面均明显优于 MS-WOA 算法，并且在两种算法对函数 F_1、F_8、F_{10} 都取到最优解情况下，由图 3、4 中对应的算法收敛曲线可知，APN-WOA 算法的收敛速度仍明显优于 MS-WOA 算法。

表 1　基准测试函数

Table 1　Benchmark functions

函数名	表达式	维度(*Dim*)	搜索区间	理论最优值
Sphere	$F_1=\sum_{i=1}^{Dim}x_i^2$	**30/200/500**	**[-100,100]**	**0**
Schwefel 2.22	$F_2=\sum_{i=1}^{Dim}\lvert x_i\rvert+\prod_{i=1}^{Dim}\lvert x_i\rvert$	**30/200/500**	**[-10,10]**	**0**
Schwefel 1.2	$F_3=\sum_{i=1}^{Dim}\left(\sum_{j-1}^{i}x_j\right)^2$	**30/200/500**	**[-100,100]**	**0**
Schwefel 2.21	$F_4=\max_i\{\lvert x_i\rvert,1\le i\le D\}$	**30/200/500**	**[-100,100]**	**0**
Rosenbrock	$F_5=\sum_{i=1}^{Dim-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	**30/200/500**	**[-30,30]**	**0**
Quartic	$F_6=\sum_{i=1}^{Dim}i\cdot x^4+random[0,1)$	**30/200/500**	**[-1.28,1.28]**	**0**
Schwefel 2.26	$F_7=\sum_{i=1}^{Dim}-x_i\sin(\sqrt{x_i})$	**30/200/500**	**[-500,500]**	$-418.9829\times Dim$
Rastrigin	$F_8=\sum_{i=1}^{Dim}[x_i^2-10\cos(2\pi x_i)+10]$	**30/200/500**	**[-5.12,5.12]**	**0**
Ackley	$F_9=-20\exp\left(-0.2\sqrt{\frac{1}{Dim}\sum_{i=1}^{Dim}x_i^2}\right)-\exp\left(\frac{1}{Dim}\sum_{i=1}^{Dim}\cos(2\pi x_i)\right)+20+e$	**30/200/500**	**[-32,32]**	**0**
Griewank	$F_{10}=\frac{1}{4000}\sum_{i=1}^{Dim}x_i^2-\prod_{i=1}^{Dim}\cos(\frac{x_i}{\sqrt{i}})+1$	**30/200/500**	**[-600,600]**	**0**
Penalized	$F_{11}=\frac{\pi}{Dim}\{10\sin(\pi y_1)+\sum_{i=1}^{Dim-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_{Dim}-1)^2\}+\sum_{I=1}^{Dim}u(x_i,10,100,4)$	**30/200/500**	**[-50,50]**	**0**
Generalized Penalized	$F_{12}=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{Dim}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_{Dim}-1)^2[1+\sin^2(2\pi x_{Dim})]\}+\sum_{i=1}^{Dim}u(x_i,5,100,4)$	**30/200/500**	**[-50,50]**	**0**

表 2　算法寻优性能比较

Table 2　Algorithm optimization performance comparison

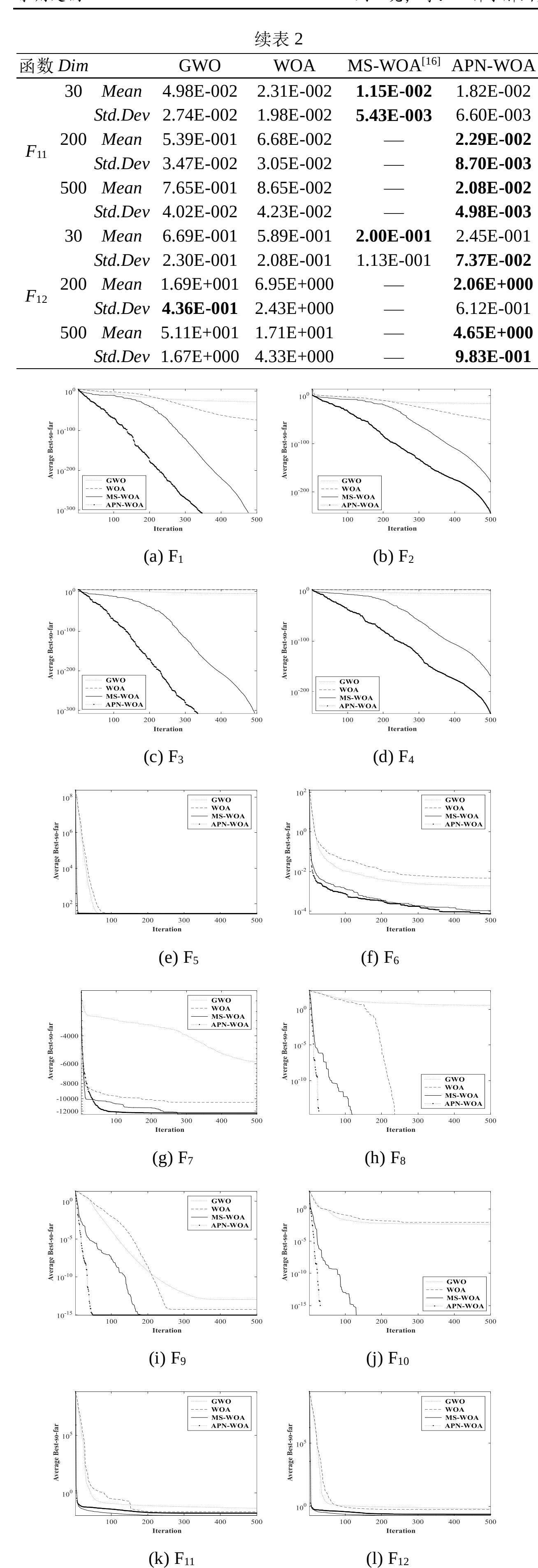
函数	*Dim*		GWO	WOA	MS-WOA[16]	APN-WOA
F_1	30	*Mean*	2.46E-027	7.95E-074	0.00E+000	**0.00E+000**
		Std.Dev	3.32E-027	3.18E-073	0.00E+000	**0.00E+000**
	200	*Mean*	1.12E-007	1.45E-072	0.00E+000	**0.00E+000**
		Std.Dev	5.39E-008	4.26E-072	0.00E+000	**0.00E+000**
	500	*Mean*	1.75E-003	6.97E-070	—	**0.00E+000**
		Std.Dev	7.84E-004	3.15E-069	—	**0.00E+000**
F_2	30	*Mean*	8.49E-017	3.32E-051	1.31E-180	**2.27E-245**
		Std.Dev	6.82E-017	1.71E-050	0.00E+000	**0.00E+000**
	200	*Mean*	2.92E-005	2.51E-048	6.59E-175	**2.61E-241**
		Std.Dev	7.37E-006	1.29E-047	0.00E+000	**0.00E+000**
	500	*Mean*	1.10E-002	1.75E-047	—	**3.52E-246**
		Std.Dev	1.90E-003	9.16E-047	—	**0.00E+000**
F_3	30	*Mean*	8.21E-006	4.43E+004	5.63E-322	**0.00E+000**
		Std.Dev	1.98E-005	1.46E+004	0.00E+000	**0.00E+000**
	200	*Mean*	2.21E+004	5.07E+006	4.74E-312	**0.00E+000**
		Std.Dev	1.20E+004	1.41E+006	0.00E+000	**0.00E+000**
	500	*Mean*	3.26E+005	3.05E+007	—	**0.00E+000**
		Std.Dev	7.66E+004	9.79E+006	—	**0.00E+000**
F_4	30	*Mean*	6.90E-007	4.15E+001	9.64E-170	**3.34E-244**
		Std.Dev	4.50E-007	2.93E+001	0.00E+000	**0.00E+000**
	200	*Mean*	2.70E+001	7.71E+001	1.12E-166	**9.22E-249**
		Std.Dev	8.71E+000	2.01E+001	0.00E+000	**0.00E+000**
	500	*Mean*	6.45E+001	8.15E+001	—	**1.33E-248**
		Std.Dev	5.11E+000	1.95E+001	—	**0.00E+000**
F_5	30	*Mean*	**2.68E+001**	2.80E+001	2.81E+001	2.78E+001
		Std.Dev	6.71E-001	3.99E-001	3.64E-001	**3.13E-001**
	200	*Mean*	1.98E+002	1.97E+02	—	**1.97E+02**
		Std.Dev	4.57E-001	1.78E-001	—	**4.78E-002**
	500	*Mean*	4.98E+002	4.96E+002	—	**4.94E+002**
		Std.Dev	2.81E-001	4.06E-001	—	**7.26E-002**

续表 2

函数	*Dim*		GWO	WOA	MS-WOA[16]	APN-WOA
F_6	30	*Mean*	1.90E-003	4.60E-003	1.05E-004	**7.25E-005**
		Std.Dev	9.78E-004	4.40E-003	7.57E-005	**6.34E-005**
	200	*Mean*	1.46E-002	3.00E-003	1.58E-004	**7.15E-005**
		Std.Dev	6.10E-003	4.10E-003	1.44E-004	**6.34E-005**
	500	*Mean*	4.62E-002	2.21E-003	—	**8.56E-005**
		Std.Dev	1.16E-002	2.57E-003	—	**9.29E-005**
F_7	30	*Mean*	-5.84E+003	-1.05E+004	-1.26E+004	**-1.23E+004**
		Std.Dev	8.88E+002	1.68E+003	**1.55E+001**	3.25E+002
	200	*Mean*	-2.79E+004	-6.97E+004	—	**-8.16E+004**
		Std.Dev	5.58E+003	1.30E+004	—	**2.88E+003**
	500	*Mean*	-5.30E+004	-1.69E+005	—	**-2.06E+005**
		Std.Dev	1.36E+004	3.17E+004	—	**7.12E+003**
F_8	30	*Mean*	3.46E+000	0.00E+000	0.00E+000	**0.00E+000**
		Std.Dev	4.52E+000	0.00E+000	0.00E+000	**0.00E+000**
	200	*Mean*	2.53E+001	0.00E+000	0.00E+000	**0.00E+000**
		Std.Dev	1.09E+001	0.00E+000	0.00E+000	**0.00E+000**
	500	*Mean*	7.66E+001	0.00E+000	—	**0.00E+000**
		Std.Dev	3.15E+001	0.00E+000	—	**0.00E+000**
F_9	30	*Mean*	1.03E-013	4.80E-015	8.88E-016	**8.88E-016**
		Std.Dev	2.03E-014	3.41E-015	0.00E+000	**0.00E+000**
	200	*Mean*	2.34E-005	4.20E-015	8.88E-016	**8.88E-016**
		Std.Dev	6.39E-006	2.46E-015	0.00E+000	**0.00E+000**
	500	*Mean*	1.82E-003	5.27E-015	—	**8.88E-016**
		Std.Dev	3.47E-004	3.05E-015	—	**0.00E+000**
F_{10}	30	*Mean*	4.10E-003	8.50E-003	0.00E+000	**0.00E+000**
		Std.Dev	7.60E-003	3.50E-002	0.00E+000	**0.00E+000**
	200	*Mean*	8.37E-003	0.00E+000	0.00E+000	**0.00E+000**
		Std.Dev	1.74E-002	0.00E+000	0.00E+000	**0.00E+000**
	500	*Mean*	2.19E-002	0.00E+000	—	**0.00E+000**
		Std.Dev	4.14E-002	0.00E+000	—	**0.00E+000**

续表 2

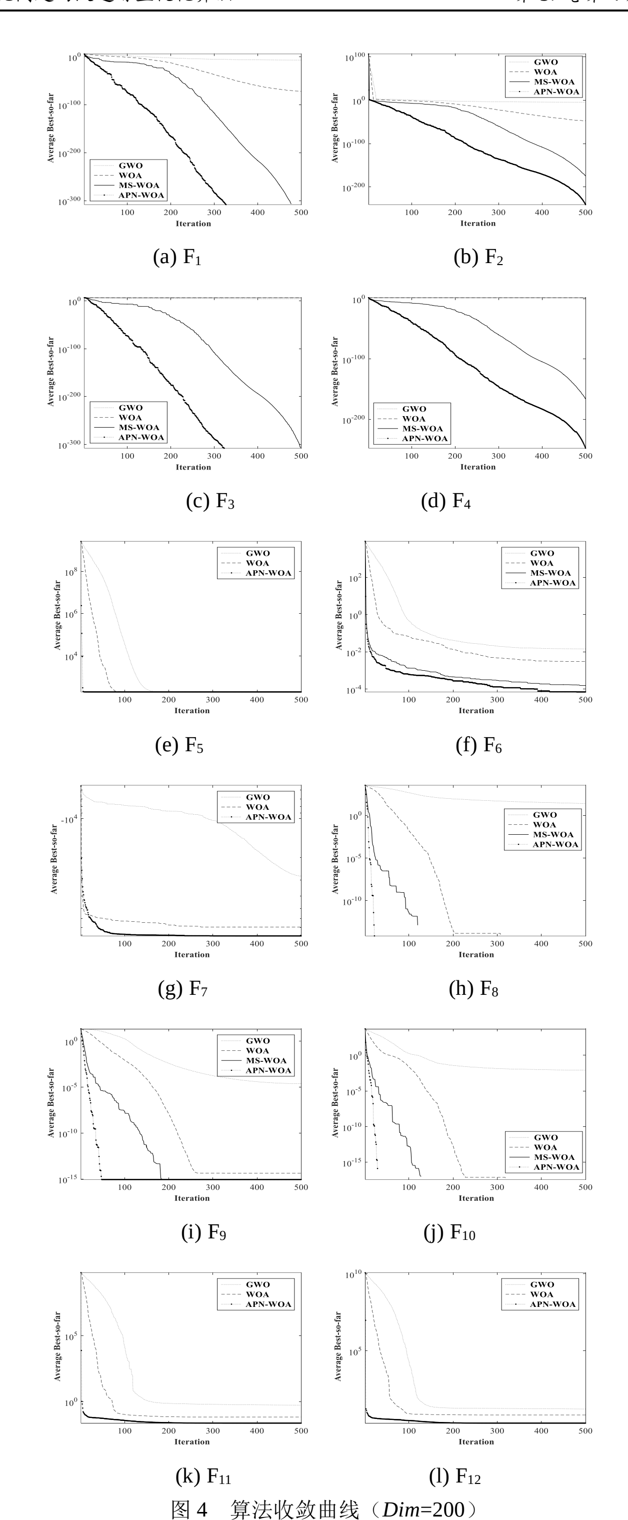
函数	Dim		GWO	WOA	MS-WOA[16]	APN-WOA
F_{11}	30	*Mean*	4.98E-002	2.31E-002	**1.15E-002**	1.82E-002
		Std.Dev	2.74E-002	1.98E-002	**5.43E-003**	6.60E-003
	200	*Mean*	5.39E-001	6.68E-002	—	**2.29E-002**
		Std.Dev	3.47E-002	3.05E-002	—	**8.70E-003**
	500	*Mean*	7.65E-001	8.65E-002	—	**2.08E-002**
		Std.Dev	4.02E-002	4.23E-002	—	**4.98E-003**
F_{12}	30	*Mean*	6.69E-001	5.89E-001	**2.00E-001**	2.45E-001
		Std.Dev	2.30E-001	2.08E-001	1.13E-001	**7.37E-002**
	200	*Mean*	1.69E+001	6.95E+000	—	**2.06E+000**
		Std.Dev	**4.36E-001**	2.43E+000	—	6.12E-001
	500	*Mean*	5.11E+001	1.71E+001	—	**4.65E+000**
		Std.Dev	1.67E+000	4.33E+000	—	**9.83E-001**



(a) F_1　(b) F_2　(c) F_3　(d) F_4　(e) F_5　(f) F_6　(g) F_7　(h) F_8　(i) F_9　(j) F_{10}　(k) F_{11}　(l) F_{12}

图 3　算法收敛曲线（Dim=30）

Fig. 3　Algorithm convergence curve（Dim=30）



(a) F_1　(b) F_2　(c) F_3　(d) F_4　(e) F_5　(f) F_6　(g) F_7　(h) F_8　(i) F_9　(j) F_{10}　(k) F_{11}　(l) F_{12}

图 4　算法收敛曲线（*Dim*=200）

Fig. 4　Algorithm convergence curve（*Dim*=200）

上述实验结果表明，本文采用的自适应概率阈值策略，使得鲸鱼在迭代过程中选择先收缩包围圈再逐步向猎物靠近的方式进行捕食，协调算法在不同迭代时期的全局探索和局部开发能力，使得鲸鱼更快地靠近猎物，从而提升算法收敛速度；同时，考虑到迭代过程中对猎物（全局最优解）的确定度会随着迭代次数增加而增加，因此，利用自适应位置权重策略，可改变猎物在迭代过程中对鲸鱼位置更新的影响力，使得鲸鱼能够更加准确地确定猎物位置，从而有效提高算法的寻优精度；最后，通过预选择小生境技术，避免鲸鱼全部聚集在某个局部最优位置造成早熟收敛的现象，保证种群多样性，从而提升算法的全局寻优能力。

综上所述，针对传统 WOA 算法收敛速度慢寻优精度低

的问题，本文采用的自适应概率阈值、自适应位置权重以及基于预选择的小生境技术三个改进策略有效地提高了算法的寻优精度和收敛速度。

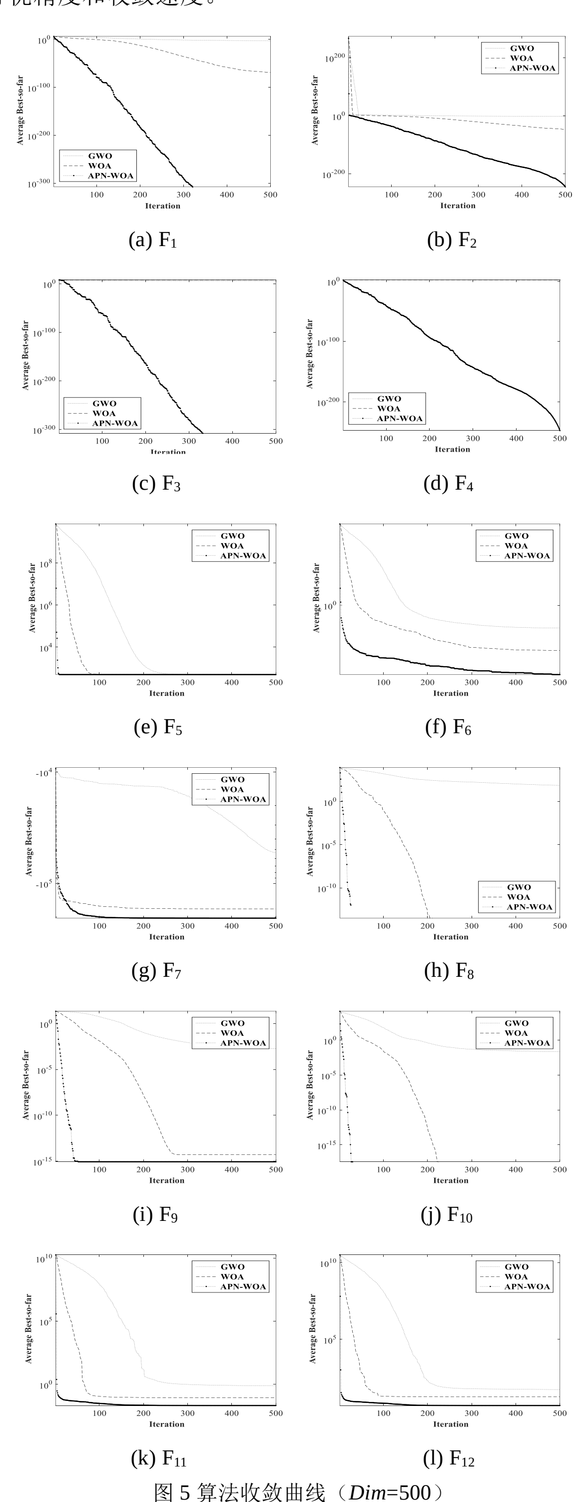


图 5 算法收敛曲线（*Dim*=500）

Fig. 5　Algorithm convergence curve（*Dim*=500）

3.2　改进策略对算法性能影响

为比较不同改进策略对 APN-WOA 算法性能的影响，设置与 3.1 节相同的实验参数，对表 1 中的 12 个基准测试函数进行优化求解，将仅采用自适应概率阈值策略的 WOA 算法记为 WOA-1，仅采用自适应位置权重策略的 WOA 算法记为 WOA-2，仅采用小生境技术的 WOA 算法记为 WOA-3，测试结果如表 3 所示。

由表 3 可知，自适应概率阈值与小生境策略对算法寻优精度的提升有限，而自适应位置权重则能有效地提升算法寻优性能，但单独的自适应位置权重策略对算法性能的提升在大部分基准测试函数上仍与结合三种改进策略的 APN-WOA 算法存在较大差距，即证明了单独自适应概率阈值与小生境策略虽无法显著地提升算法的寻优精度，但能够很好地平衡算法的全局探索和局部开发能力，以及保障种群的多样性避免算法陷入局部最优；因此 APN-WOA 综合了三种改进策略的优点，在对测试函数取相同量级的寻优精度情况下，APN-WOA 具备更快的收敛速度，且对大部分测试函数其寻优精度明显高于单一改进策略的 WOA 算法，从而证明了本文采用三种改进策略的合理性及有效性。

表 3　不同改进策略的算法比较

Table 3　Comparison of algorithms for different improvement strategies

函数		WOA-1	WOA-2	WOA-3	APN-WOA
F_1	*Mean*	7.83E-071	0.00E+000	2.41E-088	**0.00E+000**
	Std.Dev	3.05E-070	0.00E+000	9.52E-088	**0.00E+000**
F_2	*Mean*	3.26E-051	1.39E-171	3.18E-061	**4.64E-243**
	Std.Dev	8.43E-051	0.00E+000	7.32E-061	**0.00E+000**
F_3	*Mean*	4.34E+004	1.86E-294	5.13E+004	**0.00E+000**
	Std.Dev	1.67E+004	0.00E+000	1.29E+004	**0.00E+000**
F_4	*Mean*	4.44E+001	1.52E-163	8.06E+001	**5.72E-245**
	Std.Dev	3.10E+001	0.00E+000	1.10E+001	**0.00E+000**
F_5	*Mean*	**2.77E+001**	2.81E+001	2.76E+001	2.78E+001
	Std.Dev	3.44E+001	2.99E+001	4.31E+001	**2.76E+001**
F_6	*Mean*	4.30E-003	1.26E-004	2.00E-003	**1.01E-004**
	Std.Dev	3.90E-003	1.46E-004	3.30E-003	**1.26E-004**
F_7	*Mean*	-9.77E+003	**-1.25E+004**	-9.31E+003	-1.22E+004
	Std.Dev	1.89E+003	**6.97E+001**	1.03E+003	4.55E+002
F_8	*Mean*	0.00E+000	0.00E+000	0.00E+000	**0.00E+000**
	Std.Dev	0.00E+000	0.00E+000	0.00E+000	**0.00E+000**
F_9	*Mean*	4.56E-015	8.88E-016	3.84E-015	**8.88E-016**
	Std.Dev	3.02E-015	0.00E+000	2.48E-015	**0.00E+000**
F_{10}	*Mean*	7.70E-003	0.00E+000	1.85E-017	**0.00E+000**
	Std.Dev	4.20E-002	0.00E+000	6.57E-017	**0.00E+000**
F_{11}	*Mean*	3.39E-002	**1.17E-002**	1.25E+000	1.77E-002
	Std.Dev	5.55E-002	**5.50E-003**	3.09E+000	6.50E-003
F_{12}	*Mean*	5.21E-001	**1.85E-001**	6.57E-001	2.16E-001
	Std.Dev	2.73E-001	8.02E-002	2.43E-001	**6.57E-002**

3.3　与其他改进 WOA 算法性能对比

为比较 APN-WOA 与其他改进算法的性能优劣，设置种群规模为 30，最大迭代次数为 500 次，基准测试函数为 F_1、F_2、F_5、F_8、F_9、F_{10}、F_{11}，独立进行 30 次实验，引用文献[11,12]中的几组数据，比较最优解的均值以及标准差，测试结果如表 4 所示；同样的种群规模和迭代次数，采用基准测试函数 F_1~ F_{10}，独立进行 50 次实验，并引用文献[10]中的数据，与其中 WOAMC 算法进行对比，结果如表 5 所示。

由表 4、5 可知，对比文献[11,12]中所提的两种改进算法，当 IWOA 与 CWOA 对测试函数的优化求解取到理论最优解时，APN-WOA 算法同样能够对对应的测试函数取到最优解；此外，APN-WOA 算法仅在测试函数 F_5 的优化上略劣于 IWOA 算法，其余测试函数优化求解效果均优于表中两种改进算法，并且 APN-WOA 算法对函数 F_2、F_9 的优化求解的最优解标准差均取到 0，接近其理论值，对函数 F_2 求解的最优解均值高出表 3 中另两种算法多个量级；对比文献[10]中的 WOAMC 算法，APN-WOA 算法同样能对函数 F_1、F_3、F_8、F_{10} 取到理论最优解，对函数 F_2、F_4、F_9 最优解的标准

差取到 0，接近其理论值，并且 APN-WOA 对函数 F_2、F_4 求得的最优解的均值比 WOAMC 算法高出多个量级。

表 4 与其他改进算法性能对比

Table 4 Performance comparison with other improved algorithms

函数		IWOA[11]	CWOA[12]	APN-WOA
F_1	*Mean*	6.54E-125	0.00E+000	**0.00E+000**
	Std.Dev	6.80E-125	0.00E+000	**0.00E+000**
F_2	*Mean*	2.15E-073	4.56E-226	**2.27E-245**
	Std.Dev	3.64E-073	0.00E+000	**0.00E+000**
F_5	*Mean*	**2.73E+001**	2.74E+001	2.78E+001
	Std.Dev	**2.15E-001**	5.17E-000	3.13E-001
F_8	*Mean*	0.00E+000	0.00E+000	**0.00E+000**
	Std.Dev	0.00E+000	0.00E+000	**0.00E+000**
F_9	*Mean*	3.02E-015	8.88E-016	**8.88E-016**
	Std.Dev	1.95E-015	4.01E-031	**0.00E+000**
F_{10}	*Mean*	0.00E+000	0.00E+000	**0.00E+000**
	Std.Dev	0.00E+000	0.00E+000	**0.00E+000**
F_{11}	*Mean*	8.76E-002	3.09E-002	**1.82E-002**
	Std.Dev	1.20E-002	1.37E-002	**6.60E-003**

表 5 与 WOAMC 进算法性能对比

Table 5 Performance comparison with WOAMC algorithm

函数	WOAMC[10]		APN-WOA	
	Mean	*Std.Dev*	*Mean*	*Std.Dev*
F_1	0.00E+000	0.00E+000	**0.00E+000**	**0.00E+000**
F_2	3.32E-178	0.00E+000	**1.06E-241**	**0.00E+000**
F_3	0.00E+000	0.00E+000	**0.00E+000**	**0.00E+000**
F_4	1.19E-180	0.00E+000	**4.22E-247**	**0.00E+000**
F_5	2.86E+001	**1.10E-001**	**2.79E+001**	3.36E-001
F_6	1.30E-004	1.00E-004	**6.57E-005**	**5.09E-005**
F_7	-1.06E+004	2.25E+003	**-1.23E+004**	**3.79E+002**
F_8	0.00E+000	0.00E+000	**0.00E+000**	**0.00E+000**
F_9	8.88E-016	0.00E+000	**8.88E-016**	**0.00E+000**
F_{10}	0.00E+000	0.00E+000	**0.00E+000**	**0.00E+000**

综上所述，本文所提出的 APN-WOA 算法不仅相较于传统 WOA 算法在寻优精度和收敛速度方面均有显著的提升，而且，对比目前较新的改进 WOA 算法，APN-WOA 仍具有明显的优势。

4 结束语

WOA 作为一种新型启发式优化算法，它与其他元启发式优化算法相类似，在求解复杂函数优化问题时存在收敛速度慢、易陷入局部最优的问题。本文考虑到在算法迭代过程中，鲸鱼捕食策略的选择以及位置更新对算法寻优性能的影响，采用自适应概率阈值、自适应位置权重及预选择小生境技术三种改进策略对 WOA 算法进行改进。通过对 12 个基准测试函数仿真结果表明，改进算法的寻优精度及收敛速度均有大幅提升，并且相较于其他改进算法仍具有明显优势，证明了本文所提出的改进策略能够使得算法在求解复杂函数优化问题时具有更好的优化性能；而如何将改进算法应用于约束优化问题以及复杂的实际工程问题将是下一步主要研究内容。

参考文献：

[1] Mirjalili S, Lewis A. The whale optimization algorithm [J]. Advances in Engineering Software, 2016, 95: 51-67.

[2] 钟明辉，龙文．一种随机调整控制参数的鲸鱼优化算法 [J]. 科学技术与工程, 2017, 17 (12): 68-73. (Zhong Minghui, Long Wen. Whale optimization algorithm based on stochastic adjustment control parameter [J]. Science Technology and Engineering, 2017, 17 (12): 68-73.)

[3] Kennedy J, Eberhart R. Particle swarm optimization [C]// Proc of IEEE International Conference on Neural Networks. Piscataway: IEEE Press, 1995: 1942-1948.

[4] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search algorithm [J]. Information Sciences, 2009, 179 (13): 2232-2248.

[5] 沙金霞．改进鲸鱼算法在多目标水资源优化配置中的应用 [J]. 水利水电技术，2018, 49 (4): 18-26. (Sha Jinxia. Application of ameliorative whale optimization algorithm to optimal allocation of multi-objective water resources [J]. Water Resources and Hydropower Engineering, 2018, 49 (4): 18-26.)

[6] Mehne H H, Mirjalili S. A parallel numerical method for solving optimal control problems based on whale optimization algorithm [J]. Knowledge-Based Systems, 2018, 151: 114-123.

[7] Mafarja M, Mirjalili S. Whale optimization approaches for wrapper feature selection [J]. Applied Soft Computing, 2018, 62: 441-453.

[8] Abdel-Basset M, Abdle-Fatah L, Sangaiah A K. An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment [EB/OL]. (2018-01-24) [2018-10-25]. https://doi.org/10.1007/s10586-018-1769-z.

[9] Sayed G I, Darwish A, Hassanien A E. A new chaotic whale optimization algorithm for features selection [J]. Journal of Classification, 2018, 35 (2): 300-344.

[10] 郭振洲，王平，马云峰，等．基于自适应权重和柯西变异的鲸鱼优化算法 [J]. 微电子学与计算机，2017, 34 (9): 20-25. (Guo Zhenzhou, Wang Ping, Ma Yunfeng, *et al.* Whaleoptimization algorithm based on adaptive weight and cauchy mutation [J]. Microelectronics & Computer, 2017, 34 (9): 20-25.)

[11] 龙文，蔡绍洪，焦建军，等．求解大规模优化问题的改进鲸鱼优化算法 [J]. 系统工程理论与实践，2017, 37 (11): 2983-2994. (Long Wen, Cai Shaohong, Jiao Jianjun, *et al.* Improved whale optimization algorithm for large scale optimization problems [J]. Systems Engineering-Theory & Practice, 2017, 37 (11): 2983-2994.)

[12] 王坚浩，张亮，史超，等．基于混沌搜索策略的鲸鱼优化算法 [EB/OL]. (2018-08-16) [2018-10-25]. https://doi. org/10. 13195/j. kzyjc. 2018. 0098. (Wang Jianhao, Zhang Liang, Shi Chao, *et al.* Whale optimization algorithm based on chaotic search strategy [EB/OL]. (2018-08-16) [2018-10-25]. https://doi.org/10.13195/j.kzyjc.2018. 0098.)

[13] Sun Yongjun, Wang Xilu, Chen Yahuan, *et al.* A modified whale optimization algorithm for large-scale global optimization problems [J]. Expert Systems With Applications, 2018, 114: 563-577.

[14] Mafarja M M, Mirjalili S. Hybrid whale optimization algorithm with simulated annealing for feature selection [J]. Neurocomputing, 2017, 260: 302-312.

[15] 董文永，康岚兰，刘宇航，等．带自适应精英扰动及惯性权重的反向粒子群优化算法 [J]. 通信学报, 2016, 37 (12): 1-10. (Dong Wenyong, Kang Lanlan, Liu Yuhang, *et al.* Opposition-based particle swarm optimization with adaptive elite mutation and nonlinear inertia weight [J]. Journal on Communications, 2016, 37 (12): 1-10.)

[16] 何庆，魏康园，徐钦帅．基于混合策略改进的鲸鱼优化算法 [J/OL]. 计算机应用研究, 2019, 36 (12) . (2018-09-30) [2018-10-25]. http://kns. cnki. net/kcms/detail/51. 1196. TP. 20180929. 1332. 012. html. (He

Qing, Wei Kangyuan, Xu Qinshuai. Mixed strategy based improved whale optimization algorithm [J/OL]. Application Research of Computers, 2019, 36 (12) . (2018-09-30) [2018-10-25]. http://kns. cnki. Net/kcms/detail/51. 1196. TP. 20180929. 1332. 012. html.)

[17] 郑敏，高俊波．一种多模态优化的小生境遗传算法 [J]. 计算机系统应用, 2014, 23 (10): 101-106. (Zheng Min, Gao Junbo. Improved niche genetic algorithm for multimodal optimization [J]. Computer Systems & Applications, 2014, 23 (10): 101-106.)

[18] Mirjalili S, Mirjalili S M, Lewis A. Grey wolf optimizer [J]. Advances in Engineering Software, 2014, 69 (3): 46-61.